# Breaking the Barrier of 2 for the Competitiveness of Longest Queue Drop

**Antonios Antoniadis**
University of Twente, The Netherlands

**Matthias Englert**
University of Warwick, UK

**Nicolaos Matsakis**

**Pavel Veselý**
Computer Science Institute of Charles University, Prague, Czech Republic

## Abstract

We consider the problem of managing the buffer of a shared-memory switch that transmits packets of unit value. A shared-memory switch consists of an input port, a number of output ports, and a buffer with a specific capacity. In each time step, an arbitrary number of packets arrive at the input port, each packet designated for one output port. Each packet is added to the queue of the respective output port. If the total number of packets exceeds the capacity of the buffer, some packets have to be irrevocably rejected. At the end of each time step, each output port transmits a packet in its queue and the goal is to maximize the number of transmitted packets.

The Longest Queue Drop (LQD) online algorithm accepts any arriving packet to the buffer. However, if this results in the buffer exceeding its memory capacity, then LQD drops a packet from the back of whichever queue is currently the longest, breaking ties arbitrarily. The LQD algorithm was first introduced in 1991, and is known to be 2-competitive since 2001. Although LQD remains the best known online algorithm for the problem and is of practical interest, determining its true competitiveness is a long-standing open problem. We show that LQD is 1.707-competitive, establishing the first $(2-\varepsilon)$ upper bound for the competitive ratio of LQD, for a constant $\varepsilon > 0$.

**2012 ACM Subject Classification** Theory of computation → Online algorithms

**Keywords and phrases** buffer management, online scheduling, online algorithms, longest queue drop

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2021.16

**Related Version** A full version of the paper is available at `https://arxiv.org/abs/2012.03906`.

**Funding** *Antonios Antoniadis*: Work done in part while the author was at Saarland University and Max-Planck-Institute for Informatics and supported by DFG grant AN 1262/1-1.
*Pavel Veselý*: Work done while the author was at University of Warwick. Partially supported by European Research Council grant ERC-2014-CoG 647557, by GA ČR project 19-27871X, and by Charles University project UNCE/SCI/004.

## 1 Introduction

The fact that communication networks are omnipresent highlights the significance of improving their performance. A natural way to achieve such performance improvements is to develop better algorithms for buffer management of shared-memory switches which form the lower levels of network communication. We study a fundamental model of such switches.

Consider a shared-memory network switch consisting of a buffer of size $M \in \mathbb{N}$, an input port, and $N \in \mathbb{N}$ output ports. Furthermore, consider a slotted time model. In each time step, an arbitrary number of unit-valued packets arrive to the input port. Each packet comes with a label specifying the output port that it has to be forwarded to. A buffer management


algorithm has to make a decision for each packet: either irrevocably reject it, or accept it while ensuring that the buffer capacity $M$ is respected, which may mean that a previously accepted packet has to be evicted. At the end of the time step, each output port with at least one packet in the buffer destined to it transmits a packet. The goal of the buffer management algorithm is to accept/reject incoming packets or evict already accepted packets, so as to maximize the throughput, i.e., the total number of transmitted packets, while ensuring that at most $M$ packets in total are stored for all output ports at any time.

Given the inherently online nature of buffer management problems, a standard approach is to design online algorithms for them and evaluate the algorithm's performance using its competitive ratio. More specifically, an online algorithm ALG is $c$-competitive (where $c \geq 1$), if the number of packets transmitted by an optimal offline algorithm OPT (that has full knowledge of the incoming packet sequence a priori) is at most $c$ times the number of packets transmitted by ALG. There exists an extensive body of research dedicated to designing competitive online algorithms with the aim of improving the performance of networking devices that incorporate buffers (see e.g. [20, 32]).

Since packets have unit value, we can assume without loss of generality that the packets destined to a specific output port are transmitted in an earliest-arrival (FIFO) fashion and thus, it is helpful to associate each output port with a queue.

Intuitively speaking, to maximize throughput, one would like to maintain a flow of packet transmissions for as many queues in parallel as possible. It is therefore desirable to prioritize accepting packets for queues that do not have many incoming packets in the near future. Unfortunately, an online algorithm does not know which queues these are, and in order to be insured against an adversarial input it seems reasonable to try to keep the queue lengths as balanced as possible in every step. This is exactly the idea behind the online algorithm *Longest Queue Drop (LQD)*, introduced in 1991 by Wei, Coyle, and Hsiao [35]: The incoming packet is always accepted and if this causes the buffer to exceed its capacity then one packet from the longest queue, breaking ties arbitrarily, is evicted (this could be the incoming packet).[1]

The LQD algorithm, apart from being a natural online algorithm to derive, remains the only known competitive algorithm for this problem. Since the algorithm is simple and can be used, for instance, to achieve a fair distribution of the bandwidth, it is of some practical interest; see e.g. [10, 11, 12, 13, 31, 33].

### Previous Results

Hahne, Kesselman, and Mansour [21] provided the first formal analysis of LQD, showing that it is 2-competitive (see also Aiello, Kesselman, and Mansour [1]). The proof follows from a simple procedure that charges the extra profit of OPT to the profit of LQD. Furthermore, they demonstrate that LQD is at least $\sqrt{2}$-competitive, and also showed a general lower bound of $4/3$ for the competitive ratio of any deterministic online algorithm.

The analysis of LQD in [1, 21] was then refined by Kobayashi, Miyazaki, and Okabe [28] who showed that the LQD competitive ratio is at most $2 - \min_{k=1,\ldots,N}(\lfloor M/k \rfloor + k - 1)/M$. However, for $N > \sqrt{M}$, this bound becomes $2 - O(1/\sqrt{M})$ and therefore does not establish

[1] Wei, Coyle, and Hsiao proposed the LQD algorithm for the problem of shared-memory switches, consisting of $N$ input ports and $N$ output ports. Rather than assuming $N$ input ports, each of which may receive at most one packet per time step, we more generally assume that there is a single input port of infinite capacity. Furthermore, we do not put any restrictions on the number of output ports, i.e., we allow $N$ to be arbitrarily large. Again, this only makes the problem more general.

a $2-\varepsilon$ upper bound for a constant $\varepsilon > 0$ in general. Additionally, for the case of $N = 2$ output ports, LQD is exactly $\frac{4M-4}{3M-2}$-competitive [28] (we note that although this result holds for an even buffer size, the argument unfortunately breaks down when the buffer size is odd). For the case of $N = 3$ output ports, Matsakis shows that LQD is 1.5-competitive [30].

More recently, Bochkov, Davydow, Gaevoy, and Nikolenko [9] improved the lower bound on the competitiveness of LQD from $\sqrt{2}$ to approximately 1.44 (using a direct simulation of LQD and also independently, by solving a linear program). Moreover, they show that any deterministic online algorithm is at least $\sqrt{2}$-competitive, using a construction inspired by the LQD specific lower bound from [1, 21]. To the best of our knowledge, so far, no randomized algorithms for this problem have been studied.

## Our Contribution

Although LQD is the best known online algorithm for buffer management in shared-memory switches, determining its true competitiveness remains an elusive problem and has been described as a significant open problem in buffer management [20, 32]. After the initial analysis which showed that LQD is 2-competitive and not better than $\sqrt{2}$-competitive [1, 21] progress on the upper bound has been limited to special cases (e.g., with restrictions on the number of output ports or memory size) [28, 30]. In this paper, we make the first major progress in almost twenty years on upper bounding the competitive ratio of LQD. Namely, we prove the first $(2-\varepsilon)$ upper bound for a constant $\varepsilon > 0$ without restrictions on the number of ports or the size of the buffer:

▶ **Theorem 1.** *LQD is* 1.707*-competitive.*

We remark that Theorem 1 applies to LQD with any tie-breaking rule, even if tie-breaking is under control of the adversary, and that our upper bound is strictly smaller than $1+1/\sqrt{2}$.

## Our Techniques

The proof of 2-competitiveness of LQD in [1, 21] uses the following general approach. If an optimal offline algorithm OPT currently stores more packets for a queue than LQD does, these excess packets present potential extra profit for OPT. Each such potential extra packet $p$ in OPT is then matched to a packet that is transmitted by LQD at some point before packet $p$ can be transmitted by OPT.

Our approach is different in that we (for the most part) do not match specific packets to one another. Instead, the idea is to take the total profit of LQD in each step and distribute it evenly among all potential extra packets that exist at the time. As such, the scheme is less discrete than the previous one. We then carefully calculate that, for each queue, *on average* each potential extra packet in that queue receives a profit strictly larger than one.

As described here, this approach does not quite work yet. Two additional types of charging concepts have to be combined with this first idea: One involves not splitting the LQD profit completely evenly and instead slightly favoring queues with relatively few potential extra packets, and the other involves matching some of the potential extra packets of OPT to extra packets that LQD transmits. Another difficulty is that the lengths of two queues, from which packets are rejected or evicted in the same time step, may differ by one packet. This makes our proof more intricate. To deal with this, we introduce a potential function that will amortize the LQD profit in a suitable way. Then, the main challenge is to obtain useful lower bounds on the profit assigned to each queue, for which we introduce a novel scheme that relates the buffers of LQD and of OPT.

### Further Related Work

We refer the reader to the survey by Goldwasser [20] for an overview of online algorithms for buffer management problems. Additionally, the survey of Nikolenko and Kogan [32] incorporates some more recent work. In the following, we discuss some of the results related to online buffer management for switches. In general, buffer management algorithms can be partitioned into *preemptive* ones, i.e., algorithms that allow for the eviction of already accepted packets from the buffer (eviction is also referred to as preemption), and *non-preemptive* ones that never evict a packet after it has been accepted.

Kesselman and Mansour [25] study buffer management in shared-memory switches in the non-preemptive setting in which a packet has to be transmitted once it is stored in the buffer and can no longer be evicted. They introduce the Harmonic online algorithm, which tries to maintain the length of the $i^{th}$ longest queue as roughly proportional to a $1/i$ fraction of the memory. They show that this algorithm is $(\ln(N) + 2)$-competitive and give a general lower bound of $\Omega(\log N/ \log\log N)$ for the performance of any deterministic non-preemptive online algorithm. Considering the non-constant lower bound that they establish, it follows that preemption provides a significant advantage.

Eugster, Kogan, Nikolenko, and Sirotkin [19] generalize the same problem in the following two ways: First, they study unit-valued packets labeled with an output port and a processing requirement (in our case, we have a unit processing cycle per packet). Packets accepted to the same queue have the same processing requirement. They introduce the preemptive Longest-Work-Drop algorithm: If the buffer is not full, the incoming packet is accepted; otherwise, a packet is preempted from a queue that has the largest total processing requirement. They show that this algorithm is 2-competitive and at least $\sqrt{2}$-competitive and that the competitive ratio of LQD for this more general problem is at least $(\sqrt{k} - o(\sqrt{k}))$, where $k$ is the maximum processing time of any packet. Second, they address the problem of different packet values when all packets have unit processing requirements. They show that LQD is at least $(\sqrt[3]{k} - o(\sqrt[3]{k}))$-competitive in this case, where $k$ is the maximum packet value. They also introduce a new algorithm which they conjecture to have a constant competitive ratio.

Azar and Richter [6] study switches with multiple input queues. More specifically, they consider one output port and $N$ input ports and assume that each input port has an independent buffer of size $M$. At each time step, one packet can be sent from a single input port to the output port. For $M = 1$, they prove a lower bound of $1.46 - \Theta(1/N)$ for the competitive ratio of any randomized online algorithm and a lower bound of $2 - 1/N$ for deterministic online algorithms. They also give a randomized $\frac{e}{e-1} \approx 1.582$-competitive algorithm for $M > \log N$. For $M > 1$, Albers and Schmidt [3] design a deterministic 1.889-competitive algorithm for this problem and show a deterministic lower bound of $\frac{e}{e-1} \approx 1.582$ when $N \gg M$. Azar and Litichevskey [5] give a deterministic online algorithm matching this bound for large $M$.

A lot of research has been dedicated to the natural single input and single output port model. The model is trivial for unit packet values, but challenging if packets can have different values and the goal is to maximize the total value of transmitted packets. There exists a single queue for the accepted packets and one of the most studied versions of this problem requires packet transmission in the FIFO order. Kesselman, Lotker, Mansour, Patt-Shamir, Schieber, and Sviridenko [24] show that a simple greedy algorithm is exactly $(2-1/(M+1))$-competitive when preemption is allowed. A series of works gradually improved the analysis of a better online algorithm from 1.983 [26], over 7/4 [7], to $\sqrt{3}$ [17]. Kesselman, Mansour, and van Stee [26] also show a general lower bound of 1.419 for the competitive ratio of any preemptive deterministic online algorithm.

The authors of [24] introduce the bounded-delay model of single output port switches. In this model, the buffer has unlimited size and allows for packets to be transmitted in any order, however, each packet has a deadline after which it needs to be dropped from the buffer. Once again, the problem is only interesting if packets can have different values. Any deterministic online algorithm is at least $\phi \approx 1.618$-competitive [4, 15, 22, 36], and after a sequence of gradual improvements [16, 18, 29], Veselý, Chrobak, Jeż, and Sgall [34] recently gave a $\phi$-competitive algorithm. The competitive ratio of randomized algorithms is still open, with the best upper bound of $\frac{e}{e-1} \approx 1.582$ [8, 14, 23] (that holds even against the adaptive adversary), while the lower bounds are 1.25 against the oblivious adversary [8] and 4/3 against the adaptive adversary [15].

Lastly, we mention the model of Combined Input and Output Queued (CIOQ) Switches, in which the switch has $N$ input ports and $N$ output ports. Each input and output port has its own buffer and each input port can transfer a packet to any output port; however, at most one packet can be sent from any input port and at most one packet can be accepted by any output port, during one transfer cycle of the switch. A parameter $S$ called *speedup* equals the number of transfer cycles of the switch taking place per one time step. For the unit-value case, Kesselman and Rosén [27] provide a 2-competitive non-preemptive online algorithm for $S = 1$, which becomes 3-competitive for any $S$. A faster algorithm with the same competitive ratio is given by Al-Bawani, Englert, and Westermann [2].

## 2 Setup of the Analysis

We fix an arbitrary instance $I$. Let $\mathsf{OPT}$ and $\mathsf{LQD}$ be the optimal offline algorithm and the Longest Queue Drop algorithm, respectively. In a slight abuse of notation, we also denote the profit that the optimal offline algorithm gains on input instance $I$ as $\mathsf{OPT}$ and the profit that the Longest Queue Drop algorithm gains as $\mathsf{LQD}$. Our goal is to give an upper bound on $\mathsf{OPT}/\mathsf{LQD}$.

For a time step $t$ and a queue $q$, we say that $\mathsf{OPT}$ transmits an $\mathsf{OPT}$-extra packet from $q$ if $\mathsf{OPT}$ transmits a packet from $q$ in step $t$ but $\mathsf{LQD}$ does not. Equivalently, queue $q$ is non-empty in $\mathsf{OPT}$'s buffer but empty in $\mathsf{LQD}$'s buffer at $t$. Similarly, we say that $\mathsf{LQD}$ transmits an $\mathsf{LQD}$-extra packet from a queue $q$ in step $t$ if $\mathsf{LQD}$ transmits a packet from $q$ at $t$ but $\mathsf{OPT}$ does not.

Let $\mathsf{OPT}_{\mathsf{EXTRA}}$ and $\mathsf{LQD}_{\mathsf{EXTRA}}$ be the total number of transmitted $\mathsf{OPT}$-extra and $\mathsf{LQD}$-extra packets, respectively, over all time steps and queues. Then $\mathsf{OPT} - \mathsf{OPT}_{\mathsf{EXTRA}} = \mathsf{LQD} - \mathsf{LQD}_{\mathsf{EXTRA}}$ and hence $\frac{\mathsf{OPT}}{\mathsf{LQD}} = 1 + \frac{\mathsf{OPT}_{\mathsf{EXTRA}} - \mathsf{LQD}_{\mathsf{EXTRA}}}{\mathsf{LQD}}$. Therefore, if we show that $\varrho \cdot (\mathsf{OPT}_{\mathsf{EXTRA}} - \mathsf{LQD}_{\mathsf{EXTRA}}) \leq \mathsf{LQD}$ for some $\varrho > 1$, it will imply a competitive ratio of $1 + 1/\varrho < 2$ for the Longest Queue Drop algorithm.

Let $e_q$ denote the total number of transmitted $\mathsf{OPT}$-extra packets from queue $q$ over all time steps. Then we have $\mathsf{OPT}_{\mathsf{EXTRA}} = \sum_q e_q$ and we will show

$$\varrho \cdot \left( \sum_q e_q - \mathsf{LQD}_{\mathsf{EXTRA}} \right) \leq \mathsf{LQD} \ . \tag{1}$$

We now give a high-level overview of the proof of Equation (1), which consists of two parts: (i) splitting the $\mathsf{LQD}$ profit among queues $q$ with $e_q > 0$, and (ii) mapping transmitted $\mathsf{LQD}$-extra packets to queues $q$ with $e_q > 0$.

For (ii), we use the term $\mathsf{LQD}_{\mathsf{EXTRA}}$ in (1) to "cancel out" some transmitted $\mathsf{OPT}$-extra packets. To this end, we will define how each transmitted $\mathsf{LQD}$-extra packet $p$ is mapped to a queue $q$ (which is different from the one $p$ is transmitted from). Let $m_q$ be the number

of transmitted LQD-extra packets which are mapped to $q$. The mapping will be such that $\sum_q m_q \leq \mathsf{LQD}_{\mathsf{EXTRA}}$ and that $m_q \leq e_q$. Define $\hat{e}_q = e_q - m_q \geq 0$ as the number of OPT-extra packets transmitted from queue $q$ which are not canceled out.

We have $\left(\sum_q e_q - \mathsf{LQD}_{\mathsf{EXTRA}}\right) \leq \sum_q (e_q - m_q) = \sum_q \hat{e}_q$. Hence, it is sufficient for each $q$ to receive a profit of at least $\varrho \cdot \hat{e}_q$, from which it follows that $\varrho \cdot \sum_q \hat{e}_q \leq \mathsf{LQD}$, implying (1).

We describe splitting the LQD profit, enhanced with a suitable potential, in Section 3 and introduce useful quantities for bounding the profit assigned to a particular queue in Section 4. Then, in Section 5, we introduce the mapping of transmitted LQD-extra packets to queues and derive a relation between the buffers of LQD and of OPT. Finally, we put the bounds together and optimize the value of $\varrho$ in Section 6, which will yield our upper bound on the LQD competitive ratio.

## 3 Splitting the LQD Profit

In this section, we explain how the LQD profit is split. Before we proceed, we introduce some notation and terminology and define a key time step for a queue. When we refer to the state of a queue at time step $t$ under some algorithm, we refer to the state after all new packets of step $t$ have arrived and after all possible rejections/preemptions of packets by the algorithm, but before any packet is transmitted by the algorithm at the end of step $t$. (Note that by preemption we mean an eviction of an already accepted packet.) We use the following notation:

$s^t_{\mathsf{OPT}}(q)$: the number of packets in queue $q$ in the OPT buffer in step $t$,
$s^t_{\mathsf{LQD}}(q)$: the number of packets in queue $q$ in the LQD buffer in step $t$,
$s^t_{\max} = \max_q s^t_{\mathsf{LQD}}(q)$: the maximal size of a queue in the LQD buffer in step $t$.

We say that a queue $q$ is *active* in a time step $t$ if $s^t_{\mathsf{OPT}}(q) \geq 1$ or $s^t_{\mathsf{LQD}}(q) \geq 1$. Otherwise, if $q$ is empty in both buffers at $t$, we say that $q$ is *inactive* at $t$. See Figure 1 for an illustration.

### Assumptions on the Instance

First, we assume without loss of generality (w.l.o.g.) that the longest queue in LQD's buffer has at least two packets in every step before the last step when packets are transmitted by LQD.

**(A1)** We assume that once $s^t_{\max} \leq 1$, no packet arrives to any queue in any step $t' > t$. Consequently, $s^t_{\max} \geq 2$ for any step $t$ starting from the step in which the first packets are stored in the buffer and before the last step when the LQD buffer is not empty.

To see that this assumption is w.l.o.g., note that if $s^t_{\max} \leq 1$ then after packets are transmitted in step $t$, the LQD buffer is empty. Hence, LQD's processing of possible packets arriving after $t$ is essentially independent of its behavior up to step $t$, and the adversary may postpone the arrival of future packets by an arbitrary number of steps, which may only help to increase the throughput of OPT but does not change the throughput of LQD.

We also make the following assumption w.l.o.g., which will greatly reduce the additional notation required.

**(A2)** For any queue $q$ and step $t$, we assume that if $s^t_{\mathsf{LQD}}(q) \leq 1$ but at least one packet arrived to $q$ at or before time step $t$, then no packet arrives to queue $q$ after step $t$. (If $s^t_{\mathsf{LQD}}(q) = 1$ then the last packet is transmitted from $q$ in step $t$.)

To see that this assumption is w.l.o.g., we iteratively modify the instance under consideration as follows: Let $q$ be any queue $q$ that does not satisfy this assumption and let $t$ be the

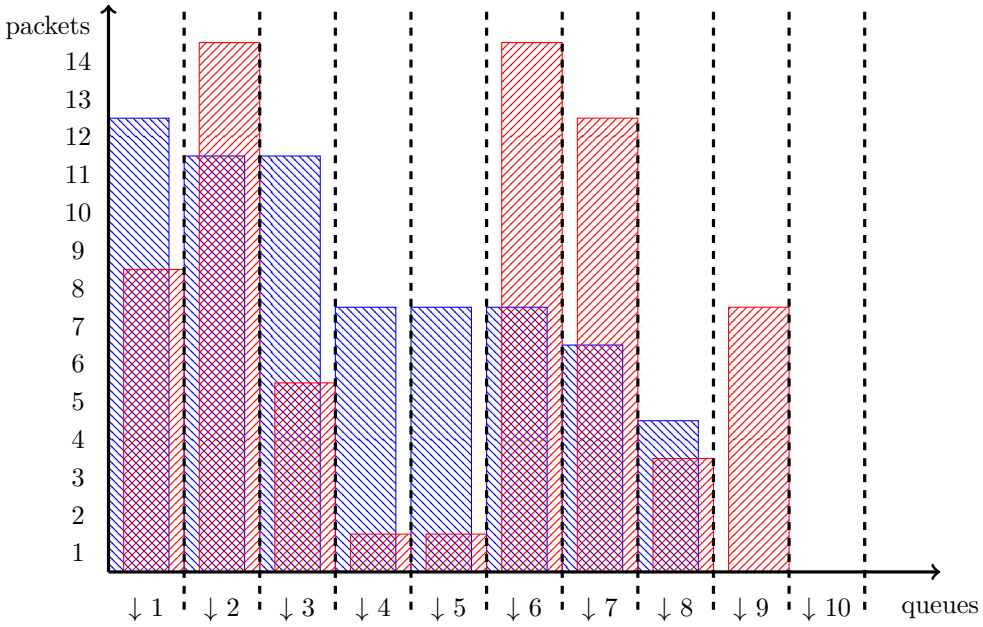


**Figure 1** An example of the buffer configuration for LQD and OPT at some time step $t$. The blue, north-west shaded areas (aligned to the left) correspond to the packets in queues of LQD and the red, north-east shaded areas (aligned to the right) to the queues of OPT. For instance, we have $s^t_{\mathsf{OPT}}(6) = 14$, and $s^t_{\mathsf{LQD}}(6) = 7$. Furthermore, $s^t_{\max} = 12$ is the maximal size of a queue for LQD. Note that an OPT-extra packet is going to be transmitted from queue 9 in step $t$, and as queue 9 is empty for LQD, no further packet arrives to this queue by assumption (A2). All the queues with an index $\geq 10$ are inactive (i.e., empty in both buffers). According to Definition 2, queues $1, 2$, and 3 overflow. As an example, assume that further 3 packets arrive into queue 8. Then LQD would first preempt a packet from queue 1 and then select two of the queues 1, 2 or 3, dropping one packet from each selected queue.

first time step such that $s^t_{\mathsf{LQD}}(q) \leq 1$ and there is a packet arriving to $q$ at $t$ or before. As $q$ does not satisfy the assumption, there is a packet arriving to $q$ after step $t$; let $p$ be the first such packet. In the modified instance, $p$ and all later packets for queue $q$ are instead sent to a new queue which is not used in the instance otherwise. Observe that the profit of LQD does not change after redirecting these packets to a new queue, while the profit of OPT cannot decrease when we make this change. We remark that the new queue is always available as the number of output ports $N$ is not restricted and can be arbitrarily large. Note that we only make this assumption to simplify our notation and it does not affect the generality of our analysis. Indeed, if the number of output ports used in the original instance is bounded by $N_0$, then after applying this transformation, there are always at most $N_0$ queues non-empty for LQD at any one time.

For instance, under assumption (A2), if an OPT-extra packet is transmitted from a queue $q$ in some step $t$ (as $q$ is empty for LQD), then no packet arrives to $q$ in any step $t' > t$.

### Overflowing Queues

Intuitively, if a packet destined to $q$ is rejected or preempted by LQD at $t$, then we say that $q$ *overflows*. Furthermore, in such a case, the LQD buffer is full in step $t$ and $q$ has $s^t_{\max} - 1$ or $s^t_{\max}$ packets at $t$ (see the example in Figure 1). This possible difference of 1 in the lengths

of two different overflowing queues makes our analysis substantially more involved.[2] For technical reasons, we also call a queue $q'$ containing at least $s^t_{\max} - 1$ packets at $t$ overflowing, provided that the LQD buffer is full and $s^t_{\mathsf{LQD}}(q') \geq 1$, even though there may be no packet for $q'$ that is rejected or preempted at time $t$ (it may even happen that no packet destined to any queue gets rejected or preempted at $t$ but there are still some overflowing queues).

▶ **Definition 2.** *We say that a queue $q$* overflows *in step $t$ if the LQD buffer is full in step $t$, $s^t_{LQD}(q) \geq s^t_{\max} - 1$, and $s^t_{LQD}(q) \geq 1$.*

Assumption (A2) also implies that once $s^t_{\mathsf{LQD}}(q) \leq 1$, then queue $q$ does not overflow after $t$ (as after step $t$, it is empty in the LQD buffer).

### Key Time Step

Based on assumption (A2), we give a definition of a key time step $t_q$ for queue $q$. For each queue $q$, we define:

$t_q$: the last time step in which queue $q$ overflows; if $q$ does not overflow in any step, we define $t_q = -1$ (we index time steps starting from 0).

Some important properties follow directly from the definition of $t_q$: No packet is ever preempted by LQD from $q$ after $t_q$ and no packet arriving to $q$ after $t_q$ is rejected by LQD, since a preemption or rejection in some step $t$ implies that the queue overflows at $t$. We remark that we define $t_q = -1$ for queues $q$ that do not overflow in any step in order to have the property that for such queues, $t_q < t$ for all time steps $t$.

We would like to keep track of how many OPT-extra packets are yet to be transmitted from a queue, for which the following notation is useful.

$e^t_q$: the number of OPT-extra packets transmitted from $q$ in step $t$ or later.

$\hat{e}^t_q = \max\{e^t_q - m_q, 0\}$: that is, $e^t_q$ adjusted for the packets that are canceled out by transmitted LQD-extra packets. Note that $m_q$ will be specified in Section 5.

Note that $e_q = e^0_q = e^{t_q}_q$ as no OPT-extra packet is transmitted before time $t_q$ by assumption (A2). Thus, $e^t_q$ is constant up to time $t_q$. After that, it further remains constant until $q$ becomes empty for LQD, and then it decreases by one in each step until it becomes equal to zero. The same property holds for $\hat{e}^t_q$. The definition of $t_q$ gives us a useful observation:

▶ **Observation 3.** *For any step $t$ and queue $q$ with $t \geq t_q$ (i.e., that does not overflow after $t$), it holds that* $\max\left\{s^t_{OPT}(q) - s^t_{LQD}(q), 0\right\} \geq e^t_q$.

### Phases

It will be convenient in certain parts of the analysis to consider time phases instead of time steps. More specifically, let $\tau_1 < \tau_2 < ... < \tau_\ell$ be the time steps in which at least one queue overflows for the last time, i.e., for each $1 \leq i \leq \ell$ there is a queue $q$ such that $\tau_i = t_q \geq 0$. Note that it has to be $\ell > 0$ so that OPT gains extra profit (equivalently, $\ell = 0$ only if $\mathsf{OPT}_{\mathsf{EXTRA}} = 0$). We call the time interval $[\tau_i, \tau_{i+1})$ the *$i$-th phase*; for $i = \ell$, we define $\tau_{\ell+1} = \infty$. We remark that time steps before $\tau_1$ do not belong to any phase, because there are no OPT-extra packets transmitted before step $\tau_1$. Finally, observe that for any queue $q$

[2] A less sophisticated version of our proof, which deals with this scenario in a less careful way, only gives an upper bound of about 1.906 on the competitive ratio. Nevertheless, this analysis still requires the majority of concepts, lemmas, and calculations developed in the paper.

that overflows at least once (i.e., $t_q \geq 0$), there has to exist an $i$ such that $t_q = \tau_i$ as this queue overflows at $t_q$ for the last time.

In the remainder of the paper, our focus will be mainly on steps $\tau_1, \ldots, \tau_\ell$. For simplicity and to avoid double indexing, we shall write $s^i_{\mathsf{LQD}}(q)$ instead of $s^{\tau_i}_{\mathsf{LQD}}(q)$, and similarly, we use index $i$ instead of $\tau_i$ in other notations. Throughout the paper, $i$ will be used solely to index phases and time steps $\tau_1, \ldots, \tau_\ell$.

### Intuition for Splitting the LQD Profit

The key ingredient of our analysis is the splitting of the LQD profit such that we assign a profit of at least $\varrho \cdot \hat{e}_q$ to each $q$. To keep track of how much profit we assigned to a queue $q$, we use counter $\Phi_q$. In particular, $\Delta^i \Phi_q$ will be the LQD profit assigned to $q$ in phase $i$ and $\Phi_q = \sum_{i=1}^{\ell} \Delta^i \Phi_q$ will be the LQD profit assigned to $q$ over all phases. We will ensure that $\mathsf{LQD} \geq \sum_q \Phi_q$. The crucial part will be to show that $\Phi_q \geq \varrho \cdot \hat{e}_q$, which, together with $\sum_q (e_q - \hat{e}_q) = \sum_q m_q \leq \mathsf{LQD}_{\mathsf{EXTRA}}$, implies (1) using

$$\mathsf{LQD} \geq \sum_q \Phi_q \geq \sum_q \varrho \cdot \hat{e}_q \geq \varrho \cdot \left( \sum_q e_q - \mathsf{LQD}_{\mathsf{EXTRA}} \right) .$$

Let $\mathsf{LQD}^i$ be the profit of LQD in phase $i$, i.e., the total number of packets transmitted by LQD in all time steps in $[\tau_i, \tau_{i+1})$. A first idea is to split $\mathsf{LQD}^i$ among queues $q$ satisfying $t_q \leq \tau_i$ proportionally to $\hat{e}^i_q$, meaning that we assign a profit of $\mathsf{LQD}^i \cdot \hat{e}^i_q / \hat{e}^i$ to a queue $q$ with $\tau_i \geq t_q$, where $\hat{e}^i = \sum_{q : \tau_i \geq t_q} \hat{e}^i_q$. Such a scheme is useful because we can relate $\hat{e}^i$ to a certain fraction of the LQD profit; this is elaborated in Section 5.

Unfortunately, this simple idea fails for "short" queues, by which we mean queues for which $\hat{e}_q$ is relatively small compared to the number of packets that LQD transmits from $q$ starting from time $t_q$. In particular, the total profit assigned to such a short queue $q$ may be very close to $\hat{e}_q$, which would only be sufficient for $\varrho = 1$, thus proving 2-competitiveness.

To give a higher profit to a short queue $q$, we choose a parameter $\alpha \in (0, 1)$ and directly assign to $q$ a $(1 - \alpha)$-fraction of the profit LQD gains by transmitting packets from $q$ itself starting at time step $t_q$, whereas the remaining $\alpha$-fraction of these packets is split proportionally to $\hat{e}^i_q$. The parameter $\alpha \approx 0.58$ is chosen at the very end of the analysis, so as to minimize the competitive ratio upper bound.

### Potential

Before describing how exactly we split the LQD profit, we introduce a potential that will help us to deal with the fact that some queues overflowing in step $t$ may only have $s^t_{\max} - 1$ packets and not $s^t_{\max}$ packets. On an intuitive level, this potential amortizes the profit assignment by moving some profit from phases with a slack to phases in which our lower bounds on the profit assigned are tight; we develop these bounds in the subsequent sections.

Namely, at any phase $i$, let $\mathcal{A}^i$ be the set of queues $q$ that are active in step $\tau_i$ and satisfy $t_q > \tau_i$ (i.e., will overflow after the beginning of phase $i$). Thus, for any such queue $q$ we have $t_q = \tau_j$ for some $j > i$ and consequently, $q$ is non-empty for LQD in every step during phase $i$ by assumption (A2) (as otherwise, $q$ would not overflow at $\tau_j$).

Then, using the aforementioned parameter $\alpha$, we define potential $\Psi^i := \alpha \cdot |\mathcal{A}^i|$. Note that the potential at the beginning is $\Psi^1 \leq \alpha \cdot M$ and after the last packet of the input instance is transmitted, the potential equals $\Psi^{\ell+1} = 0$. We define two quantities which express the change of this potential in phase $i$:

$u^i$ = the number of queues active in step $\tau_{i+1}$ that were inactive in step $\tau_i$ and will overflow after $\tau_{i+1}$, i.e., the number of "new" active queues that will overflow after $\tau_{i+1}$; and
$v^i$ = the number of queues that are active in step $\tau_i$ and overflow at $\tau_{i+1}$ for the last time, i.e., $\tau_{i+1} = t_q$ for any such queue $q$.

Let $\Delta^i\Psi := \Psi^{i+1} - \Psi^i$ be the change of the potential in phase $i$; observe that $\Delta^i\Psi = \alpha \cdot (u^i - v^i)$.

### Splitting the LQD Profit

We now formally define our scheme of splitting the LQD profit. Consider phase $i$. Let $o^i$ be the number of packets that LQD transmits in the $i$-th phase from queues $q$ with $\tau_i \geq t_q$ and $e_q > 0$, and let $n^i$ be the number of packets transmitted by LQD in phase $i$ from all other queues. Note that $\mathsf{LQD}^i = o^i + n^i$.

Apart from parameter $\alpha \in (0,1)$, we use another parameter $\beta \in (0,1)$ such that $\alpha + \beta < 1$; namely we will set $\beta = \alpha^2/(8 \cdot (1-\alpha))$ (we will require that $\alpha$ is not too close to 1 so that $\alpha + \beta < 1$). Given the two parameters, in each phase $i$, we assign an LQD profit of

$$\Delta^i\Phi_q := \underbrace{\frac{\hat{e}^i_q}{\hat{e}^i} \cdot (n^i + \alpha \cdot o^i - \Delta^i\Psi) + \beta \cdot o^i_q}_{\text{L-increase}} + \underbrace{(1-\alpha-\beta) \cdot o^i_q}_{\text{S-increase}} \tag{2}$$

to each queue $q$ with $\tau_i \geq t_q$ and $\hat{e}^i_q > 0$, where $o^i_q$ is the number of packets that LQD transmits from $q$ during the $i$-th phase.

We call the first two terms in Equation (2) (i.e., $(\hat{e}^i_q/\hat{e}^i) \cdot (n^i + \alpha \cdot o^i - \Delta^i\Psi) + \beta \cdot o^i_q$) the *L-increase* for $q$ as they will be mainly useful for "long" queues (with relatively high $\hat{e}_q$). We call the last term, $(1-\alpha-\beta) \cdot o^i_q$, the *S-increase* for $q$ as it works well for "short" queues. Note that we only assign profit to queues that already have overflown for the last time. Furthermore, once a queue $q$ with $\tau_i \geq t_q$ is empty in both the LQD and OPT buffers at the start of a phase, it does not get any profit as $\hat{e}^i_q \leq e^i_q = 0$ and $o^i_q = 0$.

To ensure feasibility of our scheme, we show that in total over all queues $q$ with $\tau_i \geq t_q$ and $\hat{e}^i_q > 0$ we assign a profit of at most $\mathsf{LQD}^i - \Delta^i\Psi$. Indeed, using $\hat{e}^i = \sum_{q:\tau_i \geq t_q} \hat{e}^i_q$ and $\sum_{q:\tau_i \geq t_q \text{ and } \hat{e}^i_q > 0} o^i_q \leq o^i$, we have

$$\begin{aligned}
\sum_{q:\tau_i \geq t_q \text{ and } \hat{e}^i_q > 0} \Delta^i\Phi_q &= \sum_{q:\tau_i \geq t_q \text{ and } \hat{e}^i_q > 0} \left( \frac{\hat{e}^i_q}{\hat{e}^i} \cdot (n^i + \alpha \cdot o^i - \Delta^i\Psi) + \beta \cdot o^i_q + (1-\alpha-\beta) \cdot o^i_q \right) \\
&\leq n^i + \alpha \cdot o^i - \Delta^i\Psi + \beta \cdot o^i + (1-\alpha-\beta) \cdot o^i \\
&= n^i + o^i - \Delta^i\Psi = \mathsf{LQD}^i - \Delta^i\Psi\,.
\end{aligned}$$

While the scheme to split the LQD profit is relatively simple to define, showing $\Phi_q \geq \varrho \cdot \hat{e}_q$ brings technical challenges, namely, in obtaining suitable lower bounds on the profits assigned proportionally to $\hat{e}^i_q$ and in summing up these lower bounds over all phases. We get our lower bound based on a novel scheme that relates the buffers of LQD and of OPT, which is introduced in the next two sections.

## 4 Live and Let Die

Analyzing the S-increases is relatively easy. Most of the remainder of the proof is focused on analyzing the L-increases. We start by deriving a helpful lower bound on $n^i + \alpha \cdot o^i - \Delta^i\Psi$. For this, we first introduce the notion of live and dying queues, which are defined with respect to a fixed queue $q$ with $t_q \leq \tau_i$ and $\hat{e}_q > 0$. For this fixed queue, we need to define live and

dying queues up until the first phase that comes after OPT transmits the last packet from $q$ not canceled out by an LQD-extra packet. Let $j_q := \min\{j : \hat{e}_q^j = 0\}$ be the index $j$ of the earliest step $\tau_j$ in which all remaining OPT-extra packets to be transmitted from $q$ are canceled out.

▶ **Definition 4.** *Fix a queue $q$ with $\hat{e}_q > 0$, and consider a phase $i$ with $t_q \leq \tau_i$ and $i \leq j_q$. Let $q'$ be a queue for which LQD stores at least one packet at time step $\tau_i$. Queue $q'$ is called* live *with respect to (w.r.t.) queue $q$ at time step $\tau_i$ if*

**(i)** *$\tau_i < t_{q'}$, i.e., $q'$ overflows at some time step after the $i$-th phase, or*
**(ii)** *$e_{q'} = 0$ and $s_{LQD}^{j_q}(q') \geq 1$,*

*or both. Otherwise, $q'$ is called* dying *with respect to queue $q$ at time $\tau_i$.*

Note that step $\tau_{j_q}$ referred to in $s_{\mathsf{LQD}}^{j_q}(q')$ is after $t_q$, since $\hat{e}_q^t = \hat{e}_q > 0$ in any step $t$ before the first OPT-extra packet is transmitted from $q$. Furthermore, $e_{q'} > 0$ implies that $q'$ becomes empty in the LQD buffer before it becomes empty in the OPT buffer, by Observation 3. Intuitively, and assuming that $\hat{e}_q = e_q$, at time step $\tau_i$, a queue $q'$ is dying with respect to $q$ if (i) it no longer overflows and (ii) either LQD runs out of packets to send from $q'$ before the time OPT does or LQD runs out of packets to send from $q'$ before the beginning of phase $j_q$.

The definition of live and dying queues implies the following property about transitions between these two types. This follows since the only property in Definition 4 (for a fixed $q$) that may change with increasing $i$ is whether or not $\tau_i < t_{q'}$.

▶ **Observation 5.** *If a queue $q'$ is dying (w.r.t. queue $q$) in time step $\tau_i$, it will never be live (w.r.t. queue $q$) in step $\tau_j$ for any $j > i$. If $q'$ is live (w.r.t. queue $q$) at $\tau_i$, it can become dying (w.r.t. queue $q$) in time step $\tau_{i+1}$ only if it overflows in time step $\tau_{i+1}$ for the last time.*

For any phase $i$, we denote the set of queues that are live in step $\tau_i$ w.r.t. $q$ as $\mathcal{L}_q^i$ and the set of queues dying in step $\tau_i$ w.r.t. $q$ as $\mathcal{D}_q^i$. For a fixed phase $i$ and queue $q$, the sets $\mathcal{L}_q^i$ and $\mathcal{D}_q^i$ partition all queues in which LQD stores packets at time $\tau_i$. Let $d_q^i$ be the number of packets transmitted from queues in $\mathcal{D}_q^i$ during phase $i$. We now relate $n^i + \alpha \cdot o^i$ to $|\mathcal{L}_q^i|$ and $d_q^i$.

▶ **Observation 6.** *It holds that $n^i \geq |\mathcal{L}_q^i| \cdot (\tau_{i+1} - \tau_i)$ and also $n^i + \alpha \cdot o^i \geq |\mathcal{L}_q^i| \cdot (\tau_{i+1} - \tau_i) + \alpha \cdot d_q^i$.*

**Proof.** Recall that $o^i$ is the number of packets that LQD transmits in phase $i$ from queues $q'$ satisfying $\tau_i \geq t_{q'}$ and $e_{q'} > 0$, and that $n^i = \mathsf{LQD}^i - o^i$ (i.e., $n^i$ is the number of packets that LQD sends in the $i$-th phase from queues $q'$ that will overflow after $\tau_i$ or that satisfy $e_{q'} = 0$). As packets sent from queues that are live in step $\tau_i$ are accounted for in $n^i$, it holds that $n^i \geq |\mathcal{L}_q^i| \cdot (\tau_{i+1} - \tau_i)$, which proves the first claim.

Since every queue $q'$ with $\tau_i \geq t_{q'}$ and $e_{q'} > 0$ is dying w.r.t. queue $q$ at time step $\tau_i$, we have that $o^i \leq d_q^i$. It holds that $|\mathcal{L}_q^i| \cdot (\tau_{i+1} - \tau_i) + d_q^i \leq \mathsf{LQD}^i = n^i + o^i$, and this inequality implies the second claim by using $o^i \leq d_q^i$ and $\alpha \leq 1$. ◀

Fix a queue $q$. We would like to lower bound the number $d_q^i$ of packets transmitted from dying queues during the $i$-th phase in some way. Note that the LQD buffer is full at times $\tau_i$ and $\tau_{i+1}$. Suppose for a moment that the set of live queues (w.r.t. queue $q$) does not change between step $\tau_i$ and step $\tau_{i+1}$, i.e., $\mathcal{L}_q^{i+1} = \mathcal{L}_q^i$. Now, if the number of packets that LQD stores in live queues $\mathcal{L}_q^i$ increases by $m$ between step $\tau_i$ and step $\tau_{i+1}$, then we know that $d_q^i \geq m$. This is because the buffer is full, so if the live queues gain $m$ packets, then dying queues must have lost at least $m$ packets (possibly more if there are new dying queues in step $\tau_{i+1}$). Since dying queues do not overflow, the only possible way to reduce the number of packets stored by LQD in dying queues is to transmit them.

We now formalize this intuition and handle cases where the set of live queues changes from one phase to the next. For the fixed queue $q$ and each phase $i$ such that $\tau_i \geq t_q$ and $i \leq j_q$, we define

$$\sigma_q^i = \begin{cases} \left(\sum_{q' \in \mathcal{L}_q^i} s_{\mathsf{LQD}}^i(q')\right) / |\mathcal{L}_q^i| & \text{if } |\mathcal{L}_q^i| \geq 1, \\ 1 & \text{otherwise}. \end{cases} \tag{3}$$

In words, $\sigma_q^i$ equals the average number of LQD packets in live queues (w.r.t. queue $q$) in step $\tau_i$, provided that there is at least one such queue. Later, in Lemma 13, we show that $\hat{e}_q^i > 0$ implies $|\mathcal{L}_q^i| \geq 1$ for any phase $i$ (i.e., that there is at least one live queue w.r.t. queue $q$) and in most cases, we will only need $\sigma_q^i$ in phases $i$ with $\hat{e}_q^i > 0$. Since live queues are non-empty for LQD, it holds that $\sigma_q^i \geq 1$. Furthermore, as the average is at most the maximum and as the maximum is an integer, this gives us the following observation.

▶ **Observation 7.** *In any phase $i$ such that $\tau_i \geq t_q$ and $i \leq j_q$, it holds that $\lceil \sigma_q^i \rceil \leq s_{\max}^i$.*

The following lower bound on $\sigma_q^i$ is useful for making $\beta$ as small as possible.

▶ **Observation 8.** *Assuming $|\mathcal{L}_q^i| \geq 1$, it holds that $\sigma_q^i \geq 2$ for any $i$ with $\tau_i \geq t_q$ and $i < j_q$.*

**Proof.** We show that any live queue $q'$ (w.r.t. queue $q$) has at least two packets in the LQD buffer in any step $\tau_i \geq t_q$ with $i < j_q$, which is sufficient as $\sigma_q^i$ is the average size of live queues, provided that $|\mathcal{L}_q^i| \geq 1$. For a live queue $q'$, consider two cases (as in Definition 4): First, if $\tau_i < t_{q'}$ then indeed $s_{\mathsf{LQD}}^i(q') \geq 2$ by assumption (A2) (if we had $s_{\mathsf{LQD}}^i(q') \leq 1$, then $q'$ would be empty at $t_{q'}$ and would not overflow in that step). Second, if $e_{q'} = 0$ and $s_{\mathsf{LQD}}^{j_q}(q') \geq 1$, where $j_q = \min\{j : \hat{e}_q^j = 0\}$, then we have that $s_{\mathsf{LQD}}^i(q') \geq 2$, using assumption (A2) again together with $i < j_q$. ◀

### Packets Transmitted from Dying Queues

We can now formally state our lower bound on the number of packets transmitted from dying queues, taking into account the change of the potential as well. As a byproduct (by rearranging the bound on $d_q^i$ below), we obtain an upper bound on $u^i$, the number of "new" active queues that will overflow after $\tau_{i+1}$, which captures the increase of the potential. Recall that $v^i$ equals the number of queues that are active in step $\tau_i$ and overflow at $\tau_{i+1}$ for the last time.

▶ **Lemma 9.** *Consider any queue $q$ with $\hat{e}_q > 0$. For each phase $i$ with $\tau_i \geq t_q$ and $\hat{e}_q^i > 0$, $d_q^i \geq (\sigma_q^{i+1} - \sigma_q^i) \cdot |\mathcal{L}_q^i| + \sigma_q^{i+1} \cdot u^i - v^i$.*

**Proof.** As $q$ is fixed, we consider live and dying queues w.r.t. queue $q$ only. For simplicity, let $\tau = \tau_i$ and $\tau' = \tau_{i+1}$, thus the $i$-th phase is $[\tau, \tau')$. By the definition of $\sigma_q^i$, live queues contain $\sigma_q^i \cdot |\mathcal{L}_q^i|$ packets in the LQD buffer in step $\tau$ and thus, dying queues $\mathcal{D}_q^i$ have $M - \sigma_q^i \cdot |\mathcal{L}_q^i|$ packets in total in step $\tau$, since the LQD buffer is full in step $\tau$. As dying queues do not overflow in any step after $\tau$, it is sufficient to show that queues $\mathcal{D}_q^i$ altogether contain at most $M - \sigma_q^i \cdot |\mathcal{L}_q^i| - (\sigma_q^{i+1} - \sigma_q^i) \cdot |\mathcal{L}_q^i| - \sigma_q^{i+1} \cdot u^i + v^i = M - \sigma_q^{i+1} \cdot |\mathcal{L}_q^i| - \sigma_q^{i+1} \cdot u^i + v^i$ packets in LQD's buffer in step $\tau'$. Let $x$ be the number of LQD packets in queues $\mathcal{D}_q^i$ in step $\tau'$, so our goal is to show

$$x \leq M - \sigma_q^{i+1} \cdot |\mathcal{L}_q^i| - \sigma_q^{i+1} \cdot u^i + v^i \,. \tag{4}$$

To this end, we analyze the LQD buffer in step $\tau'$. By Observation 5, any live queue in $\mathcal{L}_q^i$ is also live in step $\tau'$ or overflows in step $\tau'$ (possibly both). Let $\mathcal{L}' = \mathcal{L}_q^i \cup \mathcal{L}_q^{i+1}$ be

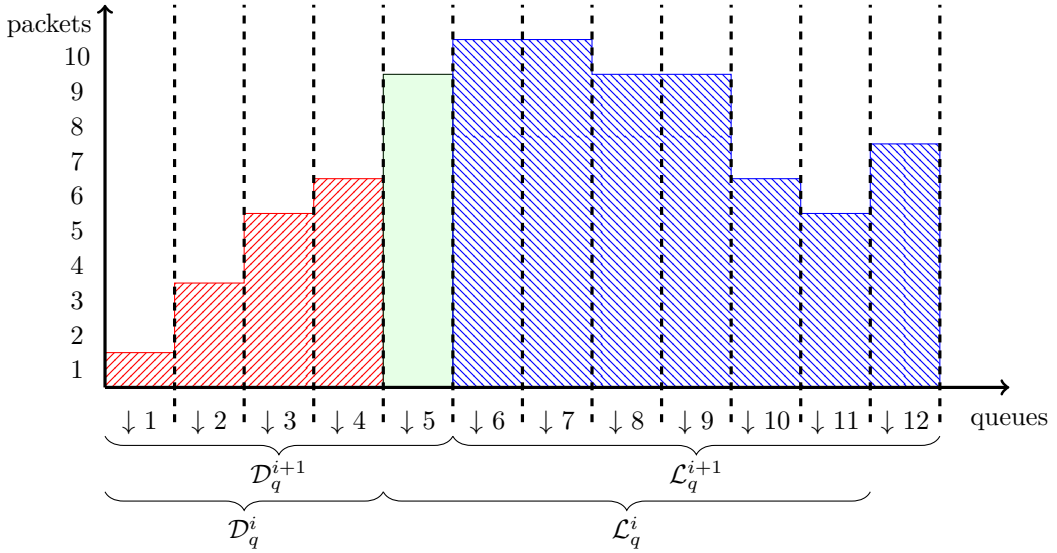


**Figure 2** An example of the LQD buffer in step $\tau' = \tau_{i+1}$ for illustrating the proof of Lemma 9. Note that $s_{\max}^{i+1} = 10$ and that queues $5-9$ overflow. However, only queue 5 becomes dying as it overflows for the last time at $\tau'$, i.e., $t_5 = \tau'$. Moreover, queue 12 was empty in step $\tau_i$ and queue 1 will become empty for LQD just after packets are transmitted in step $\tau'$ (note that no further packets will arrive to queue 1 after step $\tau'$ by assumption (A2)). We have that $\mathcal{L}' = \{5, 6, \ldots, 12\}$. Finally, $\sigma_q^{i+1} = 56/7 = 8$, since there are 56 packets in 7 live queues $\mathcal{L}_q^{i+1}$.

the set of queues that are live in step $\tau$ or in step $\tau'$. Observe that $\mathcal{L}' \cap \mathcal{D}_q^i = \emptyset$, since by Observation 5 dying queues may only become empty in LQD's buffer but not live. It follows that queues in $\mathcal{L}' \setminus \mathcal{L}_q^i$ must be inactive in step $\tau$. Next, no queue that is live in step $\tau$ is empty for LQD in step $\tau' = \tau_{i+1}$ by Definition 4, using $i < j_q$, which follows from $\hat{e}_q^i > 0$. Finally, note that $\mathcal{L}'$ may contain some dying queues in $\mathcal{D}_q^{i+1}$, but all of them must overflow at $\tau'$, and that $\mathcal{L}' \setminus \mathcal{L}_q^{i+1} \subseteq \mathcal{D}_q^{i+1} \setminus \mathcal{D}_q^i$ (however, equality is not necessarily true). Concluding, set $\mathcal{L}'$ consists of three disjoint types of queues:

- **(i)** live queues in step $\tau$ that remain live in step $\tau'$,
- **(ii)** live queues in step $\tau$ that become dying in step $\tau'$ — these are queues in $\mathcal{L}' \setminus \mathcal{L}_q^{i+1}$ and we have that $|\mathcal{L}' \setminus \mathcal{L}_q^{i+1}| = v^i$, which follows from Observation 5 and from the definition of $v^i$, and
- **(iii)** queues inactive in step $\tau$ that are live in step $\tau'$ — these are queues in $\mathcal{L}' \setminus \mathcal{L}_q^i$ and there are at least $u^i$ many of them (some live queues may not overflow in any step, so they are not accounted for in $u^i$).

See Figure 2 for an illustration.

Any queue in $\mathcal{L}' \setminus \mathcal{L}_q^{i+1}$ must overflow at $\tau'$, so it has at least $s_{\max}^{i+1} - 1 \geq \sigma_q^{i+1} - 1$ LQD packets at $\tau'$, where we use $\sigma_q^{i+1} \leq s_{\max}^{i+1}$ by Observation 7. It follows that queues in $\mathcal{D}_q^{i+1}$ have at least $x + |\mathcal{L}' \setminus \mathcal{L}_q^{i+1}| \cdot (\sigma_q^{i+1} - 1)$ packets in total in step $\tau'$. By the definition of $\sigma_q^{i+1}$ and since LQD's buffer is full at $\tau'$, queues in $\mathcal{D}_q^{i+1}$ contain $M - |\mathcal{L}_q^{i+1}| \cdot \sigma_q^{i+1}$ packets and thus $x + |\mathcal{L}' \setminus \mathcal{L}_q^{i+1}| \cdot (\sigma_q^{i+1} - 1) \leq M - |\mathcal{L}_q^{i+1}| \cdot \sigma_q^{i+1}$. Rearranging and using $|\mathcal{L}' \setminus \mathcal{L}_q^{i+1}| = v^i$, we get $x \leq M - |\mathcal{L}'| \cdot \sigma_q^{i+1} + v^i$. Using $|\mathcal{L}'| = |\mathcal{L}_q^i| + |\mathcal{L}' \setminus \mathcal{L}_q^i| \geq |\mathcal{L}_q^i| + u^i$, we obtain $x \leq M - (|\mathcal{L}_q^i| + u^i) \cdot \sigma_q^{i+1} + v^i$, implying (4). This concludes the proof as explained above. ◀

We now give two more upper bounds on $u^i$, the number of "new" active queues that will overflow after $\tau_{i+1}$. The advantage of the following bound over the one from Lemma 9 is that it does not use $\sigma_q^{i+1}$.

▶ **Lemma 10.** *Consider any queue $q$ with $\hat{e}_q > 0$. For each phase $i$ with $\tau_i \geq t_q$ and $\hat{e}_q^i > 0$, $u^i \leq \frac{1}{2} \cdot \left(|\mathcal{L}_q^i| \cdot (\sigma_q^i - 1) + d_q^i\right)$.*

**Proof.** As $q$ is fixed, we consider live and dying queues w.r.t. queue $q$ only. Let $x$ be the number of LQD packets in queues $\mathcal{D}_q^i$ in step $\tau_{i+1}$. Similarly as in the proof of Lemma 9, we show that

$$x \leq M - |\mathcal{L}_q^i| - 2 \cdot u^i \,. \tag{5}$$

This equation implies the lemma, since dying queues $\mathcal{D}_q^i$ have $M - \sigma_q^i \cdot |\mathcal{L}_q^i|$ packets in total in step $\tau_i$ and thus, $d_q^i = M - \sigma_q^i \cdot |\mathcal{L}_q^i| - x \geq M - \sigma_q^i \cdot |\mathcal{L}_q^i| - (M - |\mathcal{L}_q^i| - 2 \cdot u^i) = 2 \cdot u^i - (\sigma_q^i - 1) \cdot |\mathcal{L}_q^i|$ by (5), from which the lemma follows by rearranging. To justify (5), queues accounted for in $u^i$ are empty for LQD at $\tau_i$ and overflow after $\tau_{i+1}$, so LQD must store at least two packets in each of them in step $\tau_{i+1}$ by assumption (A2). Moreover, no queue that is live in step $\tau_i$ is empty for LQD in step $\tau_{i+1}$ by Definition 4, using $i < j_q$, which follows from $\hat{e}_q^i > 0$. Hence, there can be at most $M - |\mathcal{L}_q^i| - 2 \cdot u^i$ LQD packets in queues $\mathcal{D}_q^i$ in step $\tau_{i+1}$. ◀

Finally, we give a third upper bound on $u^i$ that is incomparable to those in Lemmas 9 and 10 and is useful for phases with a relatively small number of steps.

▶ **Lemma 11.** *For any phase $i$ with $\tau_{i+1} - \tau_i \leq \lceil \sigma_q^i \rceil - 2$, it holds that $u^i \leq \frac{1}{2} \cdot (n^i + o^i)$.*

**Proof.** Recall that the new active queues accounted for in $u^i$ are empty in both buffers in step $\tau_i$ and will overflow after $\tau_{i+1}$; let $\mathcal{U}$ be the set of these $u^i$ queues. We show that the number of packets in queues $\mathcal{U}$ in step $\tau_{i+1}$ is at most $n^i + o^i$, i.e., the number of packets LQD transmits during the $i$-th phase. This is sufficient, since any queue in $\mathcal{U}$ has at least two LQD packets at $\tau_{i+1}$ (otherwise, if there was a queue in $\mathcal{U}$ with at most one LQD packet in step $\tau_{i+1}$, no packets would arrive after $\tau_{i+1}$ to this queue by assumption (A2), so it would not overflow after $\tau_{i+1}$).

Since the LQD buffer is full in step $\tau_i$, it is sufficient to observe that any queue $q'$ has at least $s_{\mathsf{LQD}}^i(q') - (\tau_{i+1} - \tau_i)$ packets in step $\tau_{i+1}$ in the LQD buffer; in other words, that packets present in $q'$ at $\tau_i$ are not preempted till step $\tau_{i+1}$. Suppose for a contradiction that $s_{\mathsf{LQD}}^{i+1}(q') \leq s_{\mathsf{LQD}}^i(q') - (\tau_{i+1} - \tau_i) - 1$. Thus, there must be a step $t \in (\tau_i, \tau_{i+1}]$ such that a packet is preempted from $q'$ and $s_{\mathsf{LQD}}^t(q') \leq s_{\mathsf{LQD}}^i(q') - (t - \tau_i) - 1$. As $s_{\mathsf{LQD}}^i(q') \leq s_{\max}^i$, it holds that

$$s_{\mathsf{LQD}}^t(q') \leq s_{\max}^i - (t - \tau_i) - 1 \,. \tag{6}$$

Recall that there is a queue $\bar{q}$ with $\tau_i = t_{\bar{q}}$, i.e., which overflows for the last time at $\tau_i$. At $\tau_i$, queue $\bar{q}$ has at least $s_{\max}^i - 1$ packets and thus,

$$s_{\mathsf{LQD}}^t(\bar{q}) \geq s_{\max}^i - 1 - (t - \tau_i) \,. \tag{7}$$

Combining this with (6), we obtain $s_{\mathsf{LQD}}^t(\bar{q}) \geq s_{\mathsf{LQD}}^t(q')$. It holds that $t - \tau_i \leq \tau_{i+1} - \tau_i \leq \lceil \sigma_q^i \rceil - 2 \leq s_{\max}^i - 2$, where the second inequality is by the assumption of the lemma and the third inequality by Observation 7. Plugging this into (7), we obtain $s_{\mathsf{LQD}}^t(\bar{q}) \geq 1$. As a packet is preempted from $q'$ at $t$, queue $q'$ overflows at $t$, i.e., it has at least $s_{\max}^t - 1$ LQD packets. Thus, $s_{\mathsf{LQD}}^t(\bar{q}) \geq s_{\max}^t - 1$ and by Definition 2, $\bar{q}$ overflows at $t$ (here, we also use that the

LQD buffer is full at $t$ as $q'$ overflows and that $s^t_{\mathsf{LQD}}(\bar{q}) \geq 1$). However, this contradicts $t_{\bar{q}} = \tau_i < t$. Hence, any queue $q'$ has at least $s^i_{\mathsf{LQD}}(q') - (\tau_{i+1} - \tau_i)$ LQD packets in step $\tau_{i+1}$ and the number of LQD packets in queues $\mathcal{U}$ in step $\tau_{i+1}$ is at most $n^i + o^i$, which concludes the proof. ◀

## 5 Mapping Transmitted LQD-extra Packets

So far we derived a lower bound on $n^i + \alpha \cdot o^i$ for a phase $i$ which depends, among other things, on the number of queues $|\mathcal{L}^i_q|$ which are live w.r.t. a queue $q$ at time $\tau_i$. To make this bound useful, we now would like to relate $|\mathcal{L}^i_q|$ to $\hat{e}^i$.

The underlying idea behind establishing a relationship between $|\mathcal{L}^i_q|$ and $e^i = \sum_{q':\tau_i \geq t_{q'}} e^i_{q'}$ is simple. By Observation 3, quantity $e^i$ is bounded by the number of packets that are stored in OPT's buffer, but not in LQD's buffer at time $\tau_i$. Recall that the LQD buffer is full in step $\tau_i$. Intuitively, for each packet that OPT has in its buffer but LQD has not, there must be a packet that LQD has in its buffer but OPT has not. Suppose for a moment that the latter packets are all located in queues in $\mathcal{L}^i_q$. Then there can be no more than $\sigma^i_q \cdot |\mathcal{L}^i_q|$ of them and so, we would have $e^i \leq \sigma^i_q \cdot |\mathcal{L}^i_q|$. Unfortunately, things are more complicated because not all packets of the latter type may be located in live queues. We address this problem by introducing the earlier mentioned careful mapping of transmitted LQD-extra packets to cancel out some of the packets counted in $e^i$. The following notation will be useful for brevity:

- $\mathcal{Q}^i$: the set of queues $q$ with $\tau_i \geq t_q$ and $e^i_q > 0$. In words, $\mathcal{Q}^i$ is the set of queues that have overflowed for the last time by step $\tau_i$ and there are still some OPT-extra packets to be transmitted from them in phase $i$ or later.

To describe our specific mapping, we apply the procedure specified in Algorithm 1 on the solutions of LQD and OPT on the fixed instance $I$. Our values $m_q$ are given as the final values of $m'_q$ after the procedure has been run.

**Algorithm 1** Mapping Procedure

**foreach** *queue* $q$ **do**
  Initialize $m'_q := 0$ `// counter for packets assigned to` $q$
**foreach** *phase* $i$ **do**
  **foreach** *LQD-extra packet* $p$ *transmitted in phase* $i$ **do**
    **if** *there is a queue* $q \in \mathcal{Q}^i$ *with* $m'_q < e^i_q$ **then**
      $q' := \underset{q:q\in\mathcal{Q}^i \text{ and } m'_q < e^i_q}{\arg\min} \{t_q\}$ `// breaking ties arbitrarily`
      $m'_{q'} := m'_{q'} + 1$ `// assign packet` $p$ `to queue` $q'$
    `// Otherwise, packet` $p$ `is not assigned`
**foreach** *queue* $q$ **do**
  $m_q := m'_q$ `// the final value of` $m'_q$

We now show a lower bound on the $m_{q'}$ values for a phase $i$. The bound is specific to a particular queue $q \in \mathcal{Q}^i$ with $m_q < e_q$; in the following, live and dying queues are w.r.t. queue $q$. For technical reasons, we prove a lower bound on the following quantity: Let $\overline{m}^i_{q'}$ be the number of LQD-extra packets transmitted in a phase $j \geq i$ that are mapped to $q'$. The

point is that the constraint $m'_{q'} < e^i_{q'}$ for assigning an LQD-extra packet to $q'$ in Algorithm 1 implies that $e^i_{q'} \geq \overline{m}^i_{q'}$, even though it may happen that $e^i_{q'} < m_{q'}$.

For simplicity, let $z^i_q := \sum_{q' \in \mathcal{L}^i_q} [s^i_{\mathsf{OPT}}(q') > 0]$ be the number of live queues $\mathcal{L}^i_q$ that are non-empty in OPT. In words, the first term of the bound in Lemma 12 below, i.e., $\sum_{q' \in \mathcal{D}^i_q} \max\left\{s^i_{\mathsf{LQD}}(q') - s^i_{\mathsf{OPT}}(q'), 0\right\}$, equals the number of packets that LQD stores in excess of OPT in dying queues $q' \in \mathcal{D}^i_q$ with $s^i_{\mathsf{LQD}}(q') > s^i_{\mathsf{OPT}}(q')$ in step $\tau_i$, while the second term, i.e., $|\mathcal{L}^i_q| - z^i_q$, equals the number of live queues $\mathcal{L}^i_q$ that are empty in the OPT buffer in step $\tau_i$.

▶ **Lemma 12.** *For any phase $i$ and queue $q \in \mathcal{Q}^i$ with $m_q < e^i_q$ (i.e., with $\hat{e}^i_q > 0$) we have* $\sum_{q' \in \mathcal{Q}^i} \overline{m}^i_{q'} \geq \sum_{q' \in \mathcal{D}^i_q} \max\left\{s^i_{LQD}(q') - s^i_{OPT}(q'), 0\right\} + \left(|\mathcal{L}^i_q| - z^i_q\right)$ .

**Proof.** Recall from Definition 4 that $j_q = \min\{j : \hat{e}^j_q = 0\}$ is the index $j$ of the earliest step $\tau_j$ in which all remaining OPT-extra packets to be transmitted from $q$ are canceled out. Note that $i < j_q$ by the assumption of the lemma and that $m_q < e^j_q$ and $q \in \mathcal{Q}^j$ for any $j \in [i, j_q)$.

Consider a dying queue $q' \in \mathcal{D}^i_q$ with $s^i_{\mathsf{LQD}}(q') - s^i_{\mathsf{OPT}}(q') > 0$. It holds that $e_{q'} = 0$ by Observation 3. By Definition 4, $q'$ will be empty for LQD in step $\tau_{j_q}$. Since $q'$ does not overflow after time $\tau_i$ (and hence LQD will accept all packets which may arrive to $q'$ after time $\tau_i$), at least $s^i_{\mathsf{LQD}}(q') - s^i_{\mathsf{OPT}}(q')$ LQD-extra packets are transmitted from $q'$ from time $\tau_i$ until time $\tau_{j_q} - 1$, i.e., in phases $j \in [i, j_q)$. Using $m_q < e^j_q$ and $q \in \mathcal{Q}^j$ for any $j \in [i, j_q)$, all these LQD-extra packets are allocated to queues $\overline{q}$ that satisfy $t_{\overline{q}} \leq t_q \leq \tau_i$ and $e^i_{\overline{q}} > 0$; the second property holds as $t_{\overline{q}} \leq t_q \leq \tau_i$ and as $\overline{q} \in \mathcal{Q}^j$ for a phase $i \leq j < j_q$ in which the LQD-extra packet assigned to it is transmitted, by Algorithm 1. Thus, such queues $\overline{q}$ are part of the set $\mathcal{Q}^i$.

In addition, there are $|\mathcal{L}^i_q| - z^i_q$ live queues in step $\tau_i$ that are empty in the OPT buffer. Hence, LQD transmits $|\mathcal{L}^i_q| - z^i_q$ LQD-extra packets in step $\tau_i$ from such queues, and these packets are assigned to queues $\overline{q} \in \mathcal{Q}^i$ by Algorithm 1, using again that $m_q < e^i_q$. ◀

Finally, we bound the number of OPT-extra packets which are not canceled out by LQD-extra packets. Equivalently, for a queue $q$, we show a lower bound on $|\mathcal{L}^i_q|$ in terms of $\hat{e}^i$. The lemma below in particular implies that if $\hat{e}^i_q > 0$ then also $|\mathcal{L}^i_q| \geq 1$, i.e., there is at least one live queue w.r.t. queue $q$.

▶ **Lemma 13.** *For any phase $i$ and queue $q \in \mathcal{Q}^i$ with $\hat{e}^i_q > 0$, we have that* $\hat{e}^i \leq (\sigma^i_q - 1) \cdot |\mathcal{L}^i_q|$ .

**Proof.** First note that

$$\hat{e}^i = \sum_{q' \in \mathcal{Q}^i} \hat{e}^i_{q'} = \sum_{q' \in \mathcal{Q}^i} \max\{e^i_{q'} - m_{q'}, 0\} \leq \sum_{q' \in \mathcal{Q}^i} \max\{e^i_{q'} - \overline{m}^i_{q'}, 0\} = \sum_{q' \in \mathcal{Q}^i} (e^i_{q'} - \overline{m}^i_{q'}) ,$$

where the inequality holds by $\overline{m}^i_{q'} \leq m_{q'}$ and the last step follows from $e^i_{q'} \geq \overline{m}^i_{q'}$, by the definition of $\overline{m}^i_{q'}$ and Algorithm 1. Using Lemma 12, we obtain

$$\hat{e}^i \leq \sum_{q' \in \mathcal{Q}^i} (e^i_{q'}) - \sum_{q' \in \mathcal{D}^i_q} \max\left\{s^i_{\mathsf{LQD}}(q') - s^i_{\mathsf{OPT}}(q'), 0\right\} - \left(|\mathcal{L}^i_q| - z^i_q\right) \tag{8}$$

By the definition of $\sigma^i_q$ in (3) and since the LQD buffer is full in step $\tau_i$, we have

$$M = \sigma^i_q \cdot |\mathcal{L}^i_q| + \sum_{q' \in \mathcal{D}^i_q} s^i_{\mathsf{LQD}}(q') . \tag{9}$$

Regarding the OPT buffer, we have

$$M \geq \sum_{q'} s^i_{\mathsf{OPT}}(q') \geq \sum_{q' \in \mathcal{Q}^i} \max\left\{s^i_{\mathsf{OPT}}(q') - s^i_{\mathsf{LQD}}(q'), 0\right\} + \sum_{q'} \min\{s^i_{\mathsf{LQD}}(q'), s^i_{\mathsf{OPT}}(q')\}$$
$$\geq \sum_{q' \in \mathcal{Q}^i} \left(e^i_{q'}\right) + \sum_{q'} \min\{s^i_{\mathsf{LQD}}(q'), s^i_{\mathsf{OPT}}(q')\}$$
$$\geq \sum_{q' \in \mathcal{Q}^i} \left(e^i_{q'}\right) + \sum_{q' \in \mathcal{D}^i_q} \min\{s^i_{\mathsf{LQD}}(q'), s^i_{\mathsf{OPT}}(q')\} + z^i_q\,, \tag{10}$$

where the third inequality uses Observation 3. Combining (9) and (10), we obtain

$$\sum_{q' \in \mathcal{Q}^i} \left(e^i_{q'}\right) + \sum_{q' \in \mathcal{D}^i_q} \min\{s^i_{\mathsf{LQD}}(q'), s^i_{\mathsf{OPT}}(q')\} + z^i_q \leq \sigma^i_q \cdot |\mathcal{L}^i_q| + \sum_{q' \in \mathcal{D}^i_q} s^i_{\mathsf{LQD}}(q')$$

After rearranging, we get

$$\sum_{q' \in \mathcal{Q}^i} \left(e^i_{q'}\right) - \sum_{q' \in \mathcal{D}^i_q} \max\left\{s^i_{\mathsf{LQD}}(q') - s^i_{\mathsf{OPT}}(q'), 0\right\} + z^i_q \leq \sigma^i_q \cdot |\mathcal{L}^i_q|\,. \tag{11}$$

Finally, plugging (11) into Equation (8), we get $\hat{e}^i \leq \sigma^i_q \cdot |\mathcal{L}^i_q| - |\mathcal{L}^i_q|$, as desired. ◀

## 6 Putting It All Together

In this section, we complete the proof of 1.707-competitiveness for LQD. First, we use the lemmas developed in previous sections to show a lower bound on the L-increase in phase $i$ for a queue $q$. This will be divided into two cases, according to whether the value of $\sigma^i_q$ decreases or not (w.r.t. variable $i$). Next, we sum these lower bounds over all phases and derive a lower bound for this sum. Finally, we optimize the parameters $\alpha$ and $\beta$ to maximize $\varrho$ (and thus, minimize the competitive ratio upper bound) subject to $\Phi_q \geq \varrho \cdot \hat{e}_q$ for any queue $q$.

### 6.1 Lower Bounds on the L-Increase

In this section, for a queue $q$, we show lower bounds on the L-increase for a phase $i$ with $\tau_i \geq t_q$ and $\hat{e}^i_q > 0$. In such a phase, Lemma 13 implies that $\hat{e}^i \leq (\sigma^i_q - 1) \cdot |\mathcal{L}^i_q|$. We consider two main cases, depending on whether or not $\sigma_q$ decreases. We first deal with the case $\sigma^{i+1}_q \geq \sigma^i_q$.

▶ **Lemma 14.** *Consider any phase $i$ and queue $q$ with $\tau_i \geq t_q$, $\hat{e}^i_q > 0$, and $\sigma^{i+1}_q \geq \sigma^i_q$. Then the L-increase in phase $i$ for queue $q$ satisfies*

$$\frac{\hat{e}^i_q}{\hat{e}^i} \cdot (n^i + \alpha \cdot o^i - \Delta^i \Psi) + \beta \cdot o^i_q \geq \hat{e}^i_q \cdot \frac{\alpha \cdot \left(\sigma^{i+1}_q - \sigma^i_q\right) + (\tau_{i+1} - \tau_i)}{\sigma^i_q - 1}\,. \tag{12}$$

**Proof.** Recall that $d^i_q$ is the number of packets transmitted from queues in $\mathcal{D}^i_q$ during phase $i$. Using Lemma 9 and $\sigma^{i+1}_q \geq 1$, we get

$$d^i_q \geq (\sigma^{i+1}_q - \sigma^i_q) \cdot |\mathcal{L}^i_q| + \sigma^{i+1}_q \cdot u^i - v^i \geq (\sigma^{i+1}_q - \sigma^i_q) \cdot |\mathcal{L}^i_q| + u^i - v^i\,. \tag{13}$$

Multiplying (13) by $\alpha \in (0, 1)$, adding $(\tau_{i+1} - \tau_i) \cdot |\mathcal{L}^i_q|$ to both sides, and rearranging, we obtain

$$(\tau_{i+1} - \tau_i) \cdot |\mathcal{L}^i_q| + \alpha \cdot d^i_q - \alpha \cdot (u^i - v^i) \geq \left(\alpha \cdot \left(\sigma^{i+1}_q - \sigma^i_q\right) + (\tau_{i+1} - \tau_i)\right) \cdot |\mathcal{L}^i_q|.$$

Using $n^i + \alpha \cdot o^i \geq (\tau_{i+1} - \tau_i) \cdot |\mathcal{L}_q^i| + \alpha \cdot d_q^i$ by Observation 6 and $|\mathcal{L}_q^i| \geq \hat{e}^i/(\sigma_q^i - 1)$ by Lemma 13, we get $n^i + \alpha \cdot o^i - \Delta^i\Psi \geq \dfrac{\alpha \cdot \left(\sigma_q^{i+1} - \sigma_q^i\right) + (\tau_{i+1} - \tau_i)}{\sigma_q^i - 1} \cdot \hat{e}^i$, and multiplying this by $\hat{e}_q^i/\hat{e}^i$ proves (12). ◀

Next, we deal with the (most involved) case when $\sigma_q^{i+1} < \sigma_q^i$. The following technical lemma also captures the case of the last phase $i = j_q - 1$ in which we assign some LQD profit to queue $q$ (in this phase, we have $\hat{e}_q^{i+1} = 0$). We omit its proof due to space constraints.

▶ **Lemma 15.** *Consider any phase $i$ and queue $q$ with $\tau_i \geq t_q$, $\hat{e}_q^i > 0$, and $\sigma_q^{i+1} < \sigma_q^i$. Then, for $\beta = \alpha^2/(8 \cdot (1-\alpha))$ and any $0 < \alpha \leq 8/9$, it holds that*

$$\frac{\hat{e}_q^i}{\hat{e}^i} \cdot (n^i + \alpha \cdot o^i - \Delta^i\Psi) + \beta \cdot o_q^i \geq \sum_{t=\tau_i}^{\tau_{i+1}-1} \left(\frac{\hat{e}_q^t}{\sigma_q^i - 1}\right) - \frac{g_q^i}{2(\sigma_q^i - 1)} - \hat{e}_q^{i+1} \cdot \left(\frac{1}{\sigma_q^{i+1} - 1} - \frac{1}{\sigma_q^i - 1}\right), \tag{14}$$

*where $g_q^i$ is the number of steps $t \in [\tau_i, \tau_{i+1})$ with $\hat{e}_q^t > 0$ and $s_{\mathsf{LQD}}^t(q) = 0$.*

## 6.2 Total LQD Profit Assigned to a Queue

Fix a queue $q$ with $\hat{e}_q > 0$, i.e., with transmitted OPT-extra packets that are not canceled out by transmitted LQD-extra packets. We now show a lower bound on $\sum_i \Delta^i\Phi_q$. Recall that $\Delta^i\Phi_q > 0$ only for phases $i$ with $\tau_i \geq t_q$ and $\hat{e}_q^i > 0$.

First, we bound the sum of S-increases. Note that as $q$ overflows at $t_q$, LQD stores for this queue at least $s_{\max}^{t_q} - 1$ packets in step $t_q$, and since it does not overflow after $t_q$, LQD transmits at least $s_{\max}^{t_q} - 1 \geq \left\lceil \sigma_q^{t_q} \right\rceil - 1$ packets from $q$ at or after $t_q$, where the inequality is from Observation 7 (recall that $t_q = \tau_i$ for some phase $i$). Thus, the sum of S-increases is at least $(1 - \alpha - \beta) \cdot \left(\left\lceil \sigma_q^{t_q} \right\rceil - 1\right)$. The next lemma shows a bound on the total L-increase assigned to queue $q$ (its proof is omitted due to space constraints).

▶ **Lemma 16.** *Assuming $\alpha \leq 0.6$ and using $b_0 = \left\lceil \sigma_q^{t_q} \right\rceil - 1$ for simplicity, the sum of L-increases assigned to a queue $q$ with $\hat{e}_q > 0$ over all phases is at least*

$$\alpha \cdot \hat{e}_q \cdot \left(1 + \frac{b_0}{\hat{e}_q}\right) \cdot \ln\left(1 + \frac{\hat{e}_q}{b_0}\right) + \alpha \cdot \hat{e}_q \cdot \ln\left(\frac{1}{\alpha}\right).$$

Summing up the lower bound on the total S-increase with the lower bound on the total L-increase from Lemma 16, we obtain the following lower bound (where $b_0 = \left\lceil \sigma_q^{t_q} \right\rceil - 1$):

$$\sum_i \Delta^i\Phi_q \geq (1 - \alpha - \beta) \cdot b_0 + \alpha \cdot \hat{e}_q \cdot \left(1 + \frac{b_0}{\hat{e}_q}\right) \cdot \ln\left(1 + \frac{\hat{e}_q}{b_0}\right) + \alpha \cdot \hat{e}_q \cdot \ln\left(\frac{1}{\alpha}\right). \tag{15}$$

## 6.3 Calculation of the Competitive Ratio Upper Bound

According to the following lemma, we can have $\varrho = 1.41478$ in (1), which implies that the competitive ratio of LQD is at most $1 + 1/\varrho < 1.707$, according to the discussion in Section 2. Thus, the following lemma concludes the proof of Theorem 1.

▶ **Lemma 17.** *Consider a queue $q$ with $\hat{e}_q > 0$. For any values of $\sigma_q^{t_q}$ and $\hat{e}_q$, it holds that $\sum_i \Delta^i\Phi_q \geq \varrho \cdot \hat{e}_q$ for $\varrho = 1.41478$.*

**Proof sketch.** As before, let $b_0 = \left\lceil \sigma_q^{t_q} \right\rceil - 1$. Using inequality (15) as a lower bound on $\sum_i \Delta^i \Phi_q$, it is sufficient to show

$$(1-\alpha-\beta)\cdot b_0 + \alpha\cdot\hat{e}_q\cdot\left(1+\frac{b_0}{\hat{e}_q}\right)\cdot\ln\left(1+\frac{\hat{e}_q}{b_0}\right)+\alpha\cdot\hat{e}_q\cdot\ln\left(\frac{1}{\alpha}\right)\geq\varrho\cdot\hat{e}_q\,.$$

Dividing by $\hat{e}_q$ and defining $x=\frac{\hat{e}_q}{b_0}$ gives us that the optimal choice of $\varrho$ satisfies,

$$\varrho := \sup_{0<\alpha<0.6}\inf_{x>0}\left((1-\alpha-\beta)\cdot\frac{1}{x}+\alpha\cdot\left(1+\frac{1}{x}\right)\cdot\ln(1+x)+\alpha\cdot\ln\left(\frac{1}{\alpha}\right)\right)\,. \tag{16}$$

(Here, we also take into account that we require $\alpha < 0.6$ in the analysis.) Using routine calculations and optimizing through mathematical software, we get that the optimal choice for $\alpha$ is approximately 0.57635 for which $\varrho \geq 1.41478$ and therefore, the competitive ratio of LQD is at most $1 + 1/\varrho \leq 1.70683$. ◀

## References


1. W. Aiello, A. Kesselman, and Y. Mansour. Competitive buffer management for shared-memory switches. *ACM Transactions on Algorithms*, 5(1):3:1–3:16, 2008.
2. K. Al-Bawani, M. Englert, and M. Westermann. Online packet scheduling for CIOQ and buffered crossbar switches. *Algorithmica*, 80(12):3861–3888, 2018.
3. S. Albers and M. Schmidt. On the performance of greedy algorithms in packet buffering. *SIAM Journal on Computing*, 35(2):278–304, 2005.
4. N. Andelman, Y. Mansour, and A. Zhu. Competitive queueing policies for QoS switches. In *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 761–770, 2003.
5. Y. Azar and A. Litichevskey. Maximizing throughput in multi-queue switches. *Algorithmica*, 45(1):69–90, 2006.
6. Y. Azar and Y. Richter. Management of multi-queue switches in QoS networks. *Algorithmica*, 43(1-2):81–96, 2005.
7. N. Bansal, L. Fleischer, T. Kimbrel, M. Mahdian, B. Schieber, and M. Sviridenko. Further improvements in competitive guarantees for QoS buffering. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 196–207, 2004.
8. M. Bienkowski, M. Chrobak, and Ł Jeż. Randomized competitive algorithms for online buffer management in the adaptive adversary model. *Theoretical Computer Science*, 412(39):5121–5131, 2011.
9. I. Bochkov, A. Davydow, N. Gaevoy, and S. I. Nikolenko. New competitiveness bounds for the shared memory switch. *CoRR*, abs/1907.04399, 2019.
10. J. L. Bruno, B. Özden, A. Silberschatz, and H. Saran. Early fair drop: a new buffer management policy. *Multimedia Computing and Networking*, 3654:148–161, 1998.
11. S. Chamberland and B. Sansò. Overall design of reliable ip networks with performance guarantees. In *Proceedings of the IEEE International Conference on Communications: Global Convergence Through Communications (ICC)*, pages 1145–1151, 2000.
12. H. J. Chao and X. Guo. *Quality of Service Control in High-Speed Networks*. Wiley-IEEE Press, 2001.
13. H. J. Chao and B. Liu. *High Performance Switches and Routers*. Wiley-IEEE Press, 2007.
14. F. Y. L. Chin, M. Chrobak, S. P. Y. Fung, W. Jawor, J. Sgall, and T. Tichý. Online competitive algorithms for maximizing weighted throughput of unit jobs. *Journal of Discrete Algorithms*, 4(2):255–276, 2006.
15. F. Y. L. Chin and S. P. Y. Fung. Online scheduling with partial job values: Does timesharing or randomization help? *Algorithmica*, 37(3):149–164, 2003.

16 M. Chrobak, W Jawor, J. Sgall, and T. Tichý. Improved online algorithms for buffer management in QoS switches. *ACM Transactions on Algorithms*, 3(4):50, 2007.

17 M. Englert and M. Westermann. Lower and upper bounds on FIFO buffer management in QoS switches. *Algorithmica*, 53(4):523–548, 2009.

18 M. Englert and M. Westermann. Considering suppressed packets improves buffer management in quality of service switches. *SIAM Journal on Computing*, 41(5):1166–1192, 2012.

19 P. Eugster, K. Kogan, S. Nikolenko, and A. Sirotkin. Shared memory buffer management for heterogeneous packet processing. In *Proceedings of the 34th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 471–480, 2014.

20 M. H. Goldwasser. A survey of buffer management policies for packet switches. *SIGACT News*, 41(1):100–128, 2010.

21 E. L. Hahne, A. Kesselman, and Y. Mansour. Competitive buffer management for shared-memory switches. In *Proceedings of the 13th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 53–58, 2001.

22 B. Hajek. On the competitiveness of on-line scheduling of unit-length packets with hard deadlines in slotted time. In *Proceedings of the 35th Conference on Information Sciences and Systems*, pages 434–438, 2001.

23 Ł. Jeż. A universal randomized packet scheduling algorithm. *Algorithmica*, 67(4):498–515, 2013.

24 A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko. Buffer overflow management in QoS switches. *SIAM Journal on Computing*, 33(3):563–583, 2004.

25 A. Kesselman and Y. Mansour. Harmonic buffer management policy for shared memory switches. *Theoretical Computer Science*, 324(2-3):161–182, 2004.

26 A. Kesselman, Y. Mansour, and R. van Stee. Improved competitive guarantees for QoS buffering. *Algorithmica*, 43(1-2):63–80, 2005.

27 A. Kesselman and A. Rosén. Scheduling policies for CIOQ switches. *Journal of Algorithms*, 60(1):60–83, 2006.

28 K. M. Kobayashi, S. Miyazaki, and Y. Okabe. A tight bound on online buffer management for two-port shared-memory switches. In *Proceedings of the 19th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 358–364, 2007.

29 F. Li, J. Sethuraman, and C. Stein. Better online buffer management. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 199–208, 2007.

30 N. Matsakis. *Approximation Algorithms for Packing and Buffering problems*. PhD thesis, University of Warwick, UK, 2015.

31 M. Nabeshima and K. Yata. Performance improvement of active queue management with per-flow scheduling. *IEE Proceedings-Communications*, 152(6):797–803, 2005.

32 S. I. Nikolenko and K. Kogan. Single and multiple buffer processing. In *Encyclopedia of Algorithms*, pages 1988–1994. Springer, 2016.

33 B. Suter, T. V. Lakshman, D. Stiliadis, and A. K. Choudhury. Design considerations for supporting TCP with per-flow queueing. In *Proceedings of the 17th IEEE Conference on Computer Communications (INFOCOM)*, pages 299–306, 1998.

34 P. Veselý, M. Chrobak, Ł. Jeż, and J. Sgall. A $\phi$-competitive algorithm for scheduling packets with deadlines. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 123–142, 2019.

35 S. X. Wei, E. J. Coyle, and M. T. Hsiao. An optimal buffer management policy for high-performance packet switching. In *Proceedings of the Global Communication Conference (GLOBECOM)*, pages 924–928, 1991.

36 A. Zhu. Analysis of queueing policies in QoS switches. *Journal of Algorithms*, 53(2):137–168, 2004.